CHAP 3 :

STRUCTURES HIÉRARCHIQUES - ARBRES –

Université Sétif 1
Faculté des sciences
Département d'informatique

Algorithmique et Structures de Données

2017-2018 / Dr. L.Douidi

Arbre - Introduction

- Les structures de données que nous avons vu jusqu'ici (tableaux, listes, piles, files) sont *linéaires*, dans le sens où elles stockent les éléments les uns à la suite des autres : on peut les représenter comme des éléments placés sur une ligne, ou des oiseaux sur un fil électrique.

Arbre - **Introduction**

- La structure d'arbre est très utilisée en informatique. D'une part parce que les informations sont souvent hiérarchisées, et peuvent être représentées naturellement sous une forme arborescente, et d'autre part, parce que les structures de données arborescentes permettent de stocker des données volumineuses de façon que leur accès soit efficace.

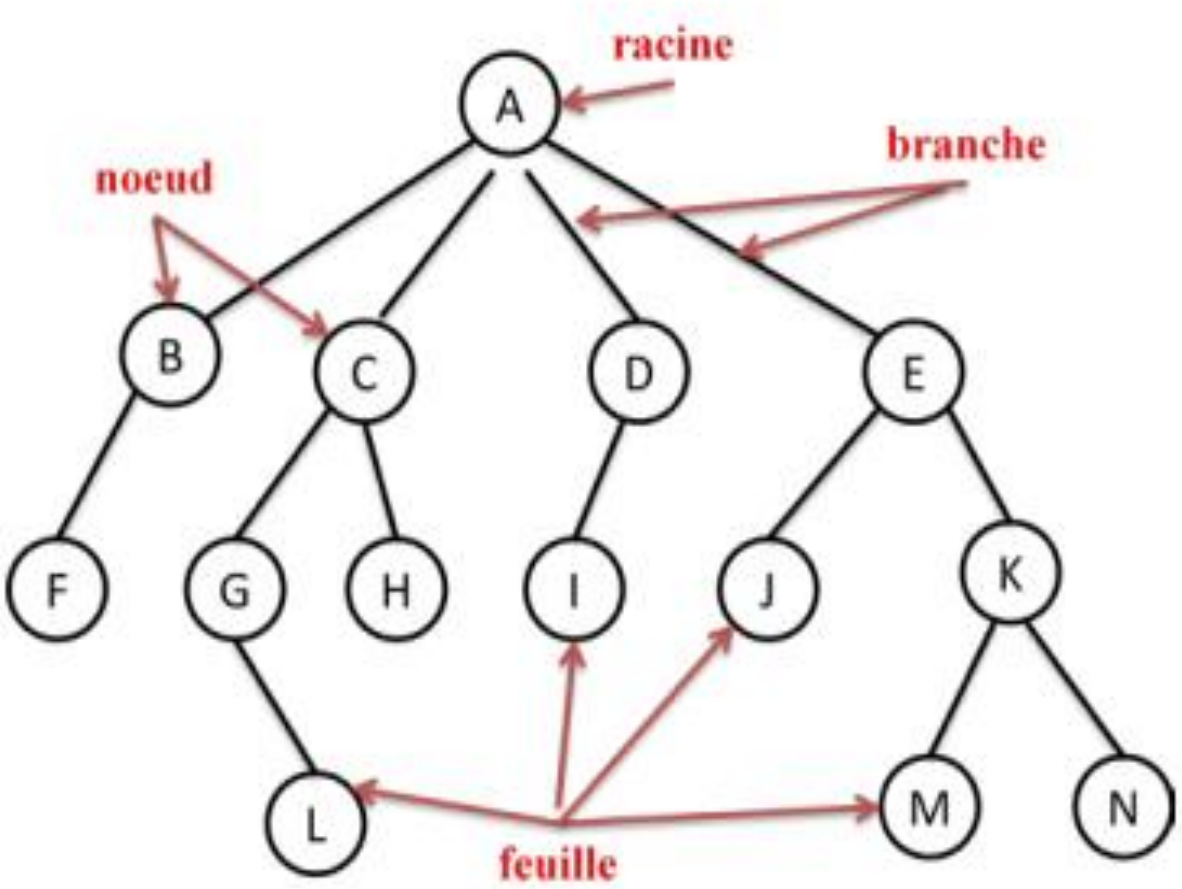
La complexité des algorithmes d'insertion de suppression ou de recherche est généralement plus faible que dans le cas des listes.

Idée-clé: Façon d'organiser les informations de sorte à y accéder rapidement.

Arbre - **Définition**

- Un arbre est une structure de données (souvent dynamique) représentant un ensemble de valeurs organisées hiérarchiquement.
- Chaque valeur est stockée dans un nœud.
- Les nœuds sont connectés entre eux par des relations parent/fils. (branche)
- A part le nœud racine, tous les autres nœuds ont exactement un seul nœud parent et zéro ou plusieurs nœuds fils.
- Le nœud racine n'a pas de parent et possède zéro ou plusieurs fils.
- Les nœuds qui n'ont pas de fils sont appelés *feuilles* (ou nœuds *externes*), les autres (ceux qui ont au moins un fils) sont appelés nœuds *internes*.

Arbre - schéma



A: **Racine**

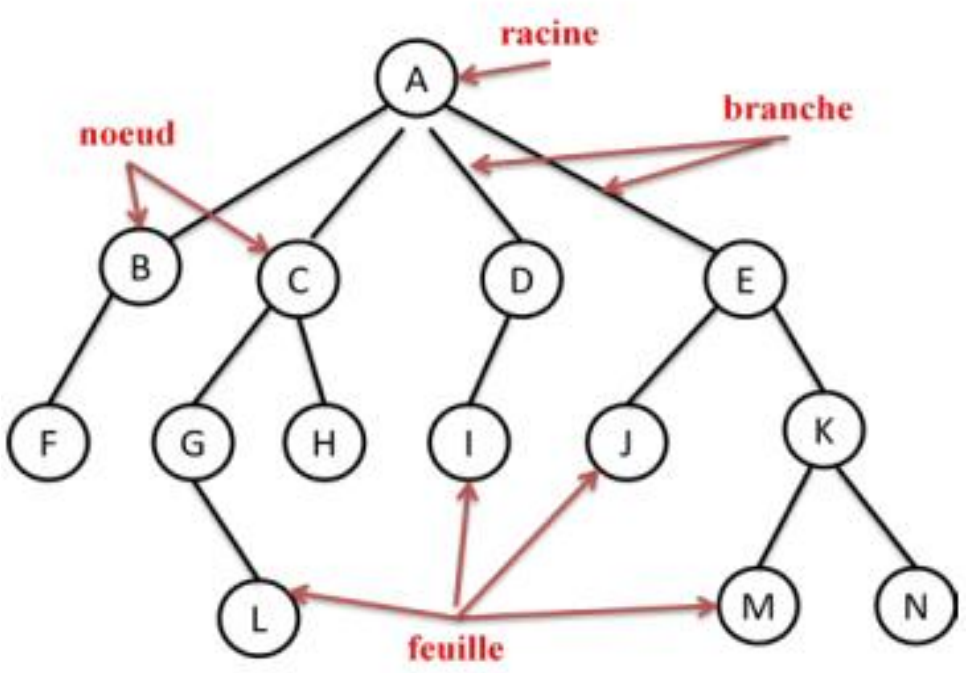
B,C,G,D,E,K: **nœuds internes**

F,L,H,I,J,M,N : **feuilles**

terminologie

- **Etiquette :** ou nom du sommet représente la **"valeur"** du nœud ou bien **l'information** associée au nœud.
  - Un arbre étiqueté par A, B,…, N ou 1,2,…

Les fils d'un nœud sont les racines de ses sous-arbres; sur l'exemple, les fils de A sont B, C, D et E;
Le père d'un nœud x autre que la racine est l'unique nœud dont x est un fils;
Sur l'exemple, C est le père de G et H ; la racine d'un arbre n'a pas de père ;



terminologie

- **Un nœud interne** est un nœud qui a au moins un fils; sur l'exemple, D est un nœud interne;
- **une feuille** d'un arbre est un nœud sans fils; sur l'exemple, F, L, H, I, J , M et N sont des feuilles;
- **Les ancêtres** d'un nœud **a** sont les nœuds qui sont sur le chemin entre la racine (incluse) et **a** (inclus) ; **les Ancêtres de a différents de a** sont les ancêtres propres de a; sur l'exemple, les ancêtres de K sont K, E et A;
- **les descendants** d'un nœud **a** sont les nœuds qui appartiennent au sous-arbre de racine **a**; **les descendants de a différents de a** sont les descendants propres de a ; sur l'exemple, les descendants de E sont E, J, K , M et N.
- **Les frères** d'un nœud **A** sont les nœuds qui possèdent le même père que A ; sur l'exemple, G et H sont frères.

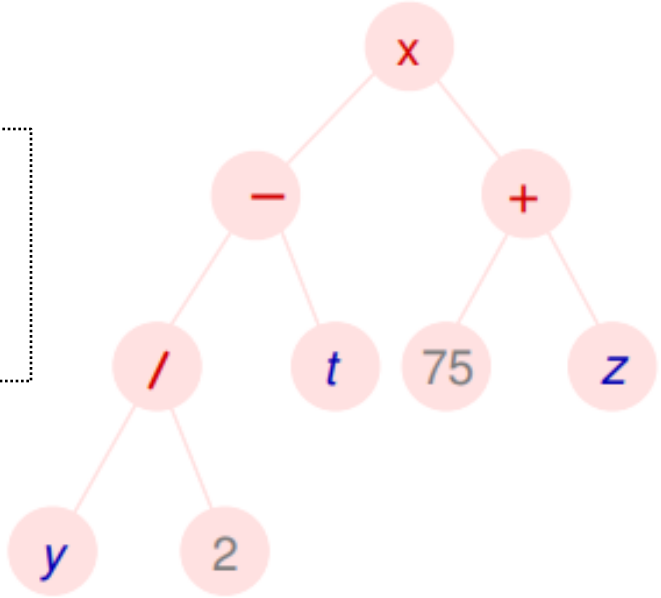
Quelques exemples de données arborescentes

- ***Expressions arithmétiques:***

On peut représenter les expressions arithmétiques par des arbres étiquetés par des opérateurs, des constantes et des variables. La structure de l'arbre rend compte de la priorité des opérateurs et rend inutile tout parenthèsage.

L'arbre correspondant à l'expression:

(y/2) – t) x (75+z)



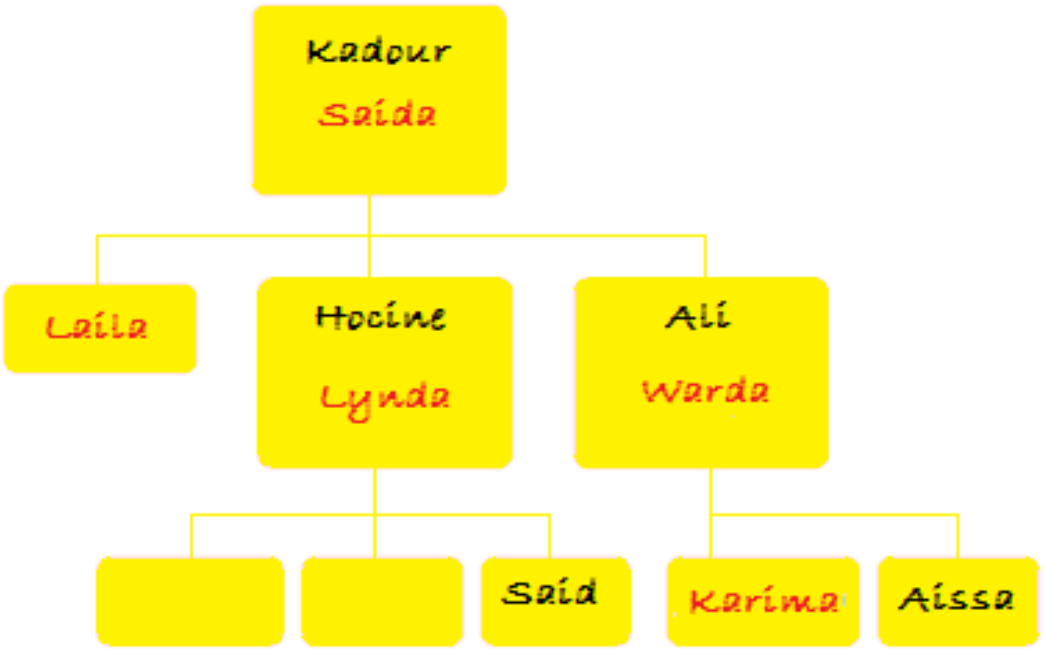
Exercice : Dessinez l'arbre correspondant à l'expression 3(2x − 1) + 1.

Quelques exemples de données arborescentes

Arbres syntaxiques **:** Un arbre syntaxique représente l'analyse d'une phrase à partir d'un ensemble de règles qui constitue la grammaire : une phrase est composée d'un groupe nominal suivi d'un groupe verbal, un groupe nominal peut-ˆêtre constitué´e d'un article et d'un nom commun,. . .

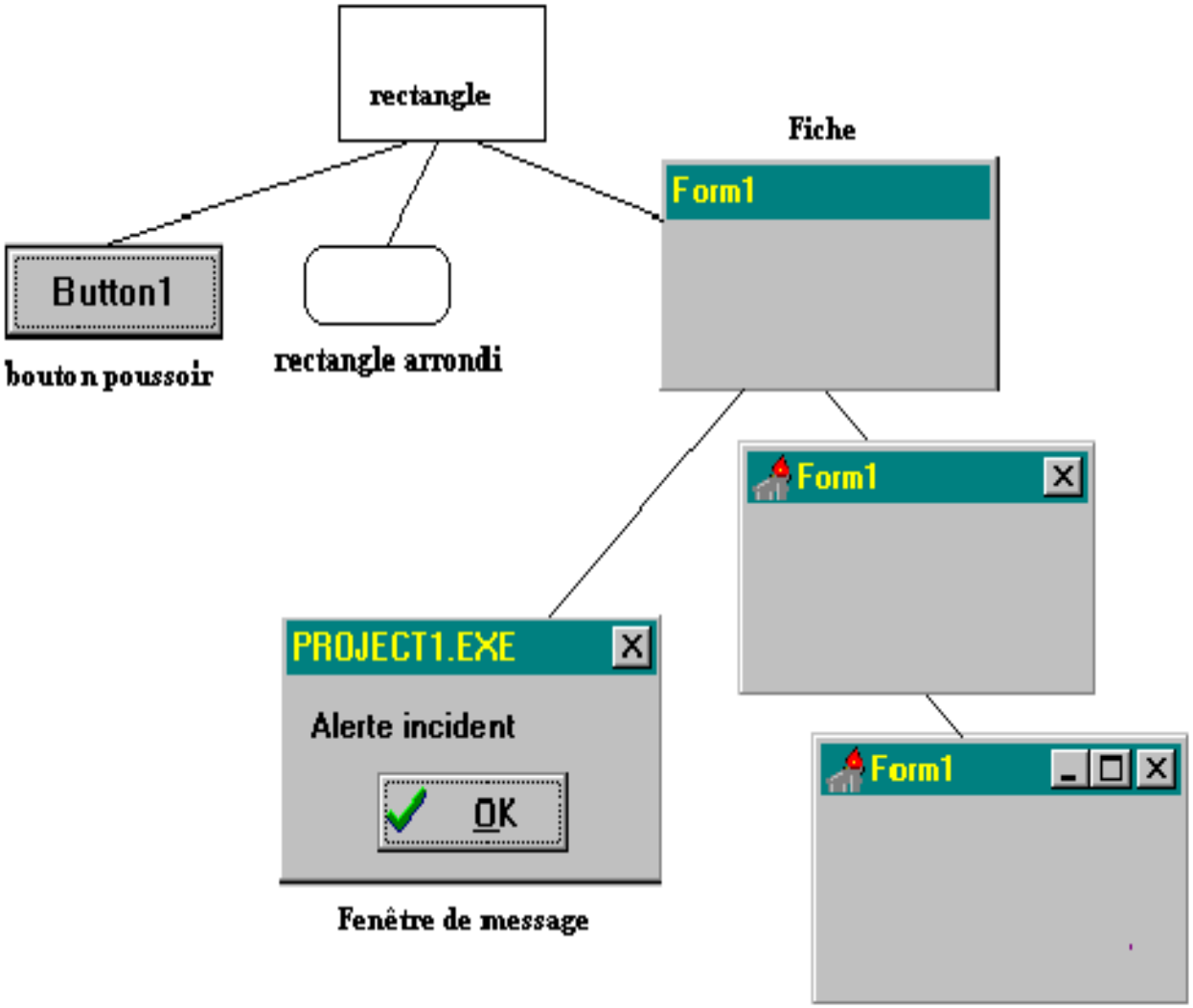
Quelques exemples de données arborescentes

- ***Arbres généalogiques*** **:** Un arbre généalogique (descendant dans le cas présent) représente la descendance d'une personne ou d'un couple. Les nœuds de l'arbre sont ´étiquetés par les membres de la famille et leurs conjoints. L'arborescence est construite à partir des liens de parenté (les enfants du couple).



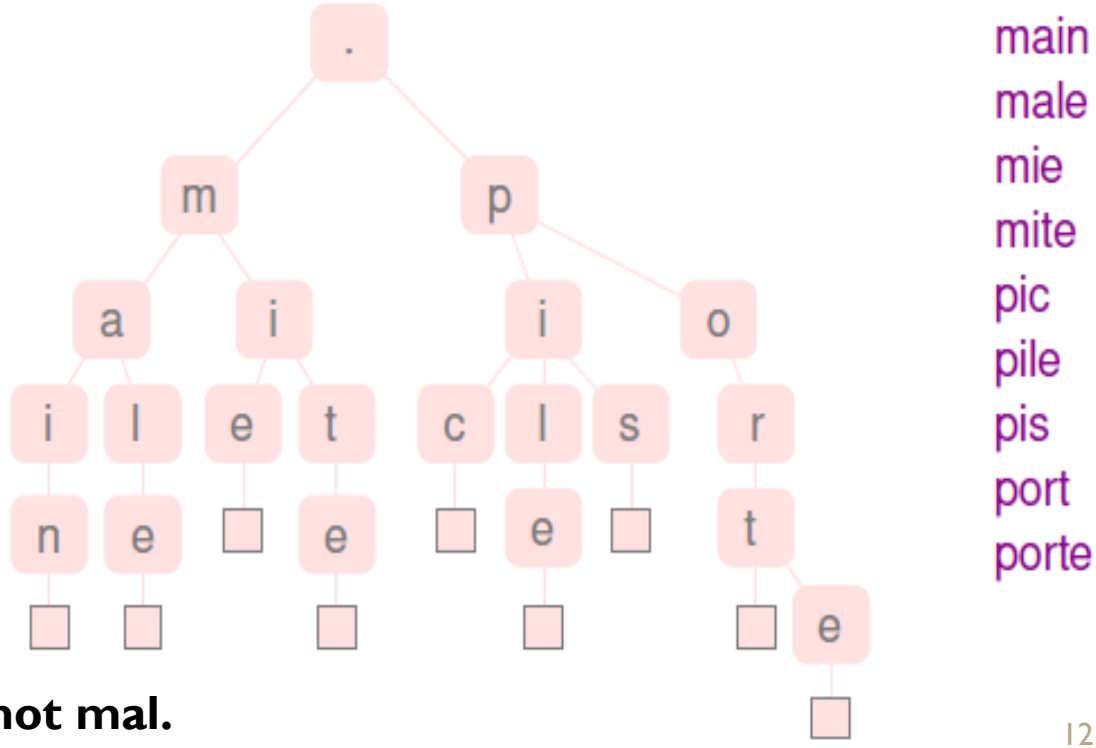
Quelques exemples de données arborescentes

- Arbre **d'héritage**



Quelques exemples de données arborescentes

- ***Arbre lexicographique*** : Un arbre lexicographique, ou arbre en parties communes, ou dictionnaire, représente un ensemble de mots. Les préfixes communs à plusieurs mots apparaissent une seule fois dans l'arbre, ce qui se traduit par un gain d'espace mémoire. De plus la recherche d'un mot est assez efficace, puisqu'il suffit de parcourir une branche de l'arbre en partant de la racine, en cherchant a chaque niveau parmi les fils du nœud courant la lettre du mot de rang correspondant.



Exercice : Rajoutez le mot mal.

Quelques exemples de données arborescentes

- arborescence des répertoires d'un ordinateur,

Quelques exemples de données arborescentes

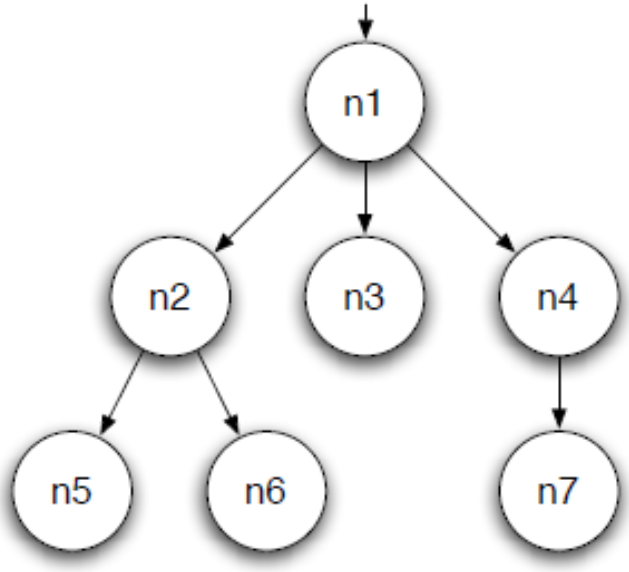
- hiérarchie des rubriques d'un site ou fils des réponses dans un forum

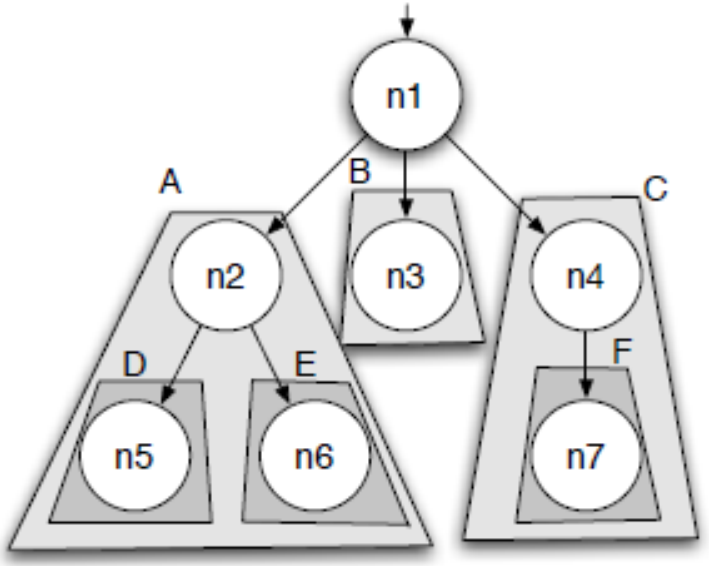
Arbre – Définition récursive

- Il est possible de donner deux types de définitions des arbres :
  - une définition « classique », en termes d'ensembles (mentionnée précédemment);
  - ou bien une définition récursive.
- Suivant la définition qu'on adopte, on pourra formuler les algorithmes de façon itérative ou récursive.

Arbre – Définition récursive

- **Cas particulier:**
  NULL est un arbre (l'arbre vide, contenant zéro nœud)
- **Cas général:** si T est un nœud et si T1, T2, ...Tm sont des arbres, alors on peut construire un nouvel arbre en connectant T1, T2, ...Tm comme des fils à T. chaque Ti est définit de la même manière (récursivement). T1, T2, ... Tm sont alors des sous-arbres de T.



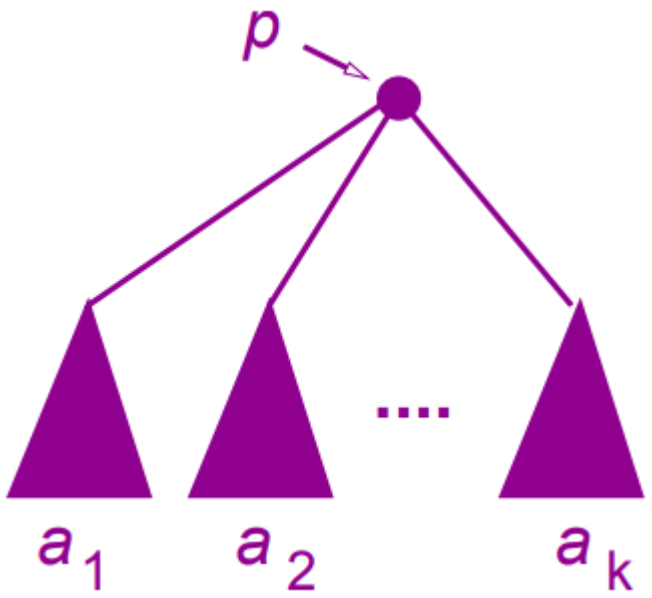


La vue récursive de l'arbre de la Figure a gauche

Arbre – Définition récursive

- Un arbre est constitué
  - d'un nœud *p*, sa **racine**,
  - d'une suite de **sous-arbres** ($a_1, a_2, \ldots, a_k$).

  Les racines des arbres $a_1, a_2, \ldots, a_k$ sont les fils de *p*

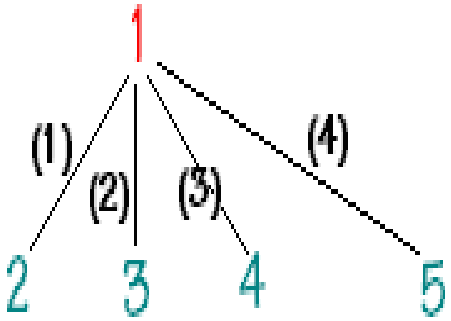


Vocabulaire

- **Degré d'un nœud :**

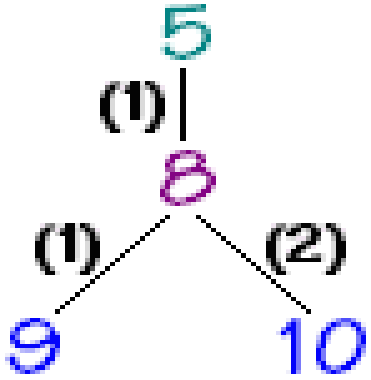
Par définition le **degré** d'un nœud est égal au **nombre de ses descendants** (enfants/fils)

Le nœud 1 est de degré 4, car il a 4 enfants/fils



Le nœud 5 n'ayant qu'un enfant son degré est 1.

Le nœud 8 est de degré 2 car il a 2 enfants.

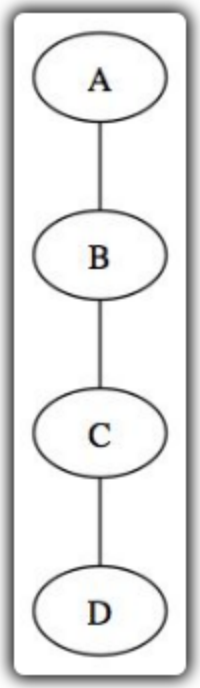


- **Taille d'un arbre**

On appelle **taille** d'un arbre le nombre total de nœuds de cet arbre
Un arbre vide est de taille égale à 0.
Exemple: Taille arbre 1 =5; taille arbre 2 = 4

arbre dégénéré

- **Remarquons** que lorsqu'un arbre a **tous ses nœuds de degré 1**, on le nomme **arbre dégénéré** et que c'est en fait une **liste**.



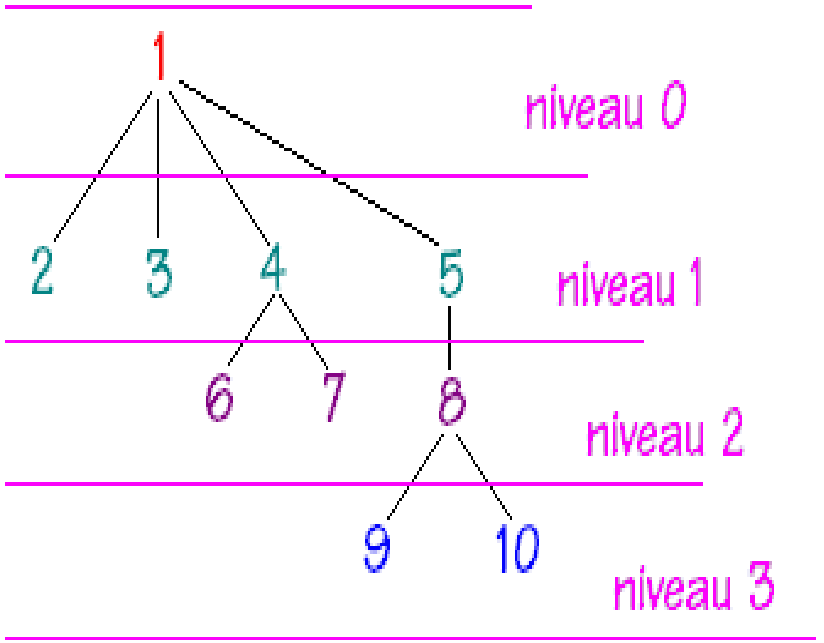
Arbre – hauteur d'un nœud

- **Hauteur, profondeur ou niveau d'un nœud**

Nous conviendrons de définir la **hauteur**(ou **profondeur** ou **niveau**) d'un nœud X comme égale **au nombre de nœuds à partir de la racine** pour aller jusqu'au nœud X.

Soit **h** la fonction hauteur d'un nœud :
Pour atteindre le nœud étiqueté 9, il faut parcourir le lien 1--5, puis 5--8, puis enfin 8--9 soient 4 nœuds donc 9 est de profondeur ou de hauteur égale à 4, soit **h(9) = 4**.

h(7) = 3.



NB: L'arbre vide à une hauteur 0, et
L'arbre réduit à une racine a une hauteur 1

Arbre – Chemin d'un nœud

- On appelle chemin du nœud X la **suite des nœuds** par lesquels il faut passer pour aller de la racine vers le nœud X

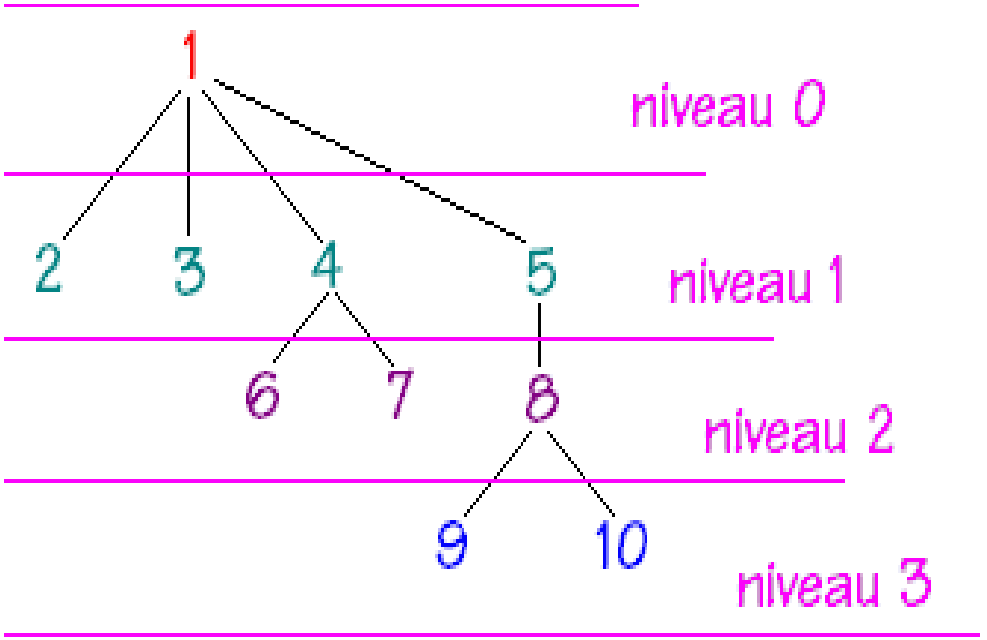
Chemin du nœud 9 = (1,5,8,9)

.....

Chemin du nœud 7 = (1,4,7)

Chemin du nœud 5 = (1,5)

Chemin du nœud 1 = (1)



h(X) = NbrNoeud(Chemin(X)).

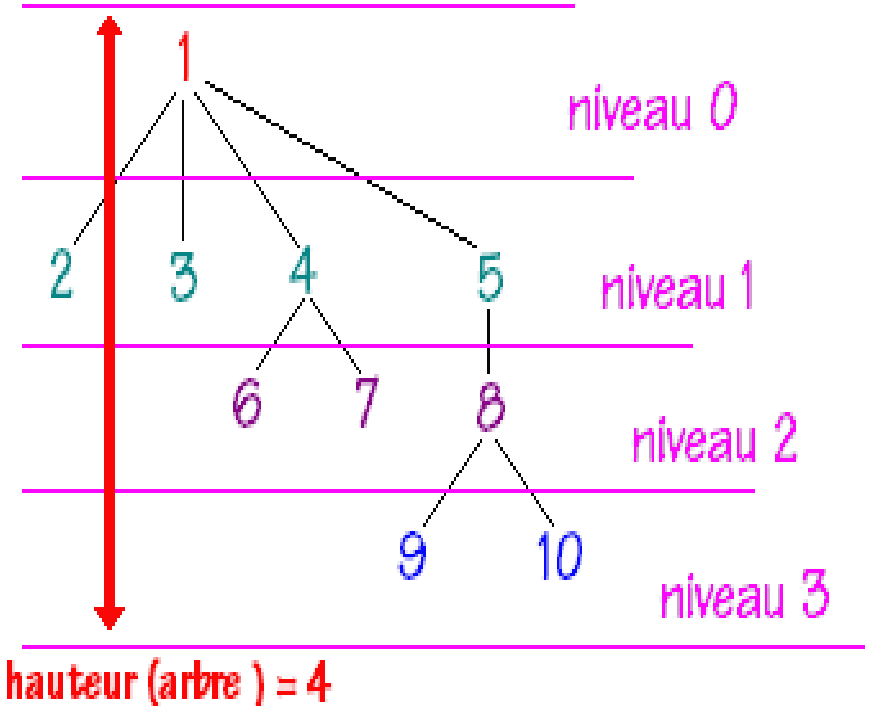
Arbre – hauteur d'un Arbre

- C'est le **nombre de nœuds du chemin le plus long** dans l'arbre. La hauteur **h** d'un arbre correspond donc au nombre de niveau maximum :

- **h (Arbre) = max { h (X) / ∀ X, X nœud de Arbre }**
  si Arbre = Ø alors h(Arbre) = 0

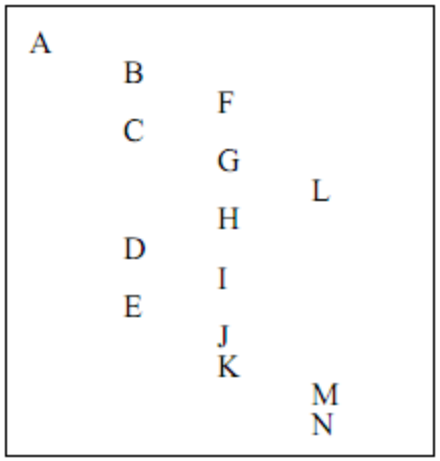
Calculons récursivement la hauteur du nœud 9, notée h(9) :

h(9) = 1+h(8)
h(8) = 1+h(5)
h(5) = 1+h(1)
h(1) = 1 => h(5)=2 => h(8)=3 => h(9)=4

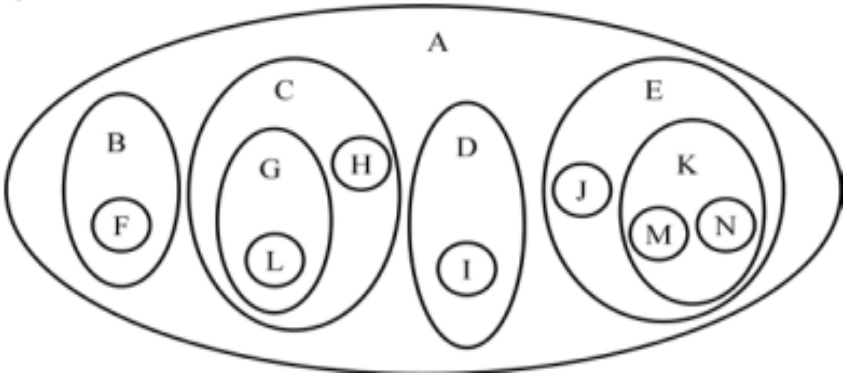


Représentation des arbres

- **Graphe :** c'est la représentation la plus utilisée comme déjà vue.
- **Indentation :**



- **Ensemble imbriqué**

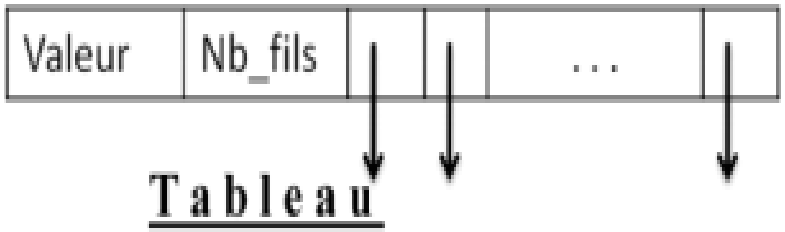


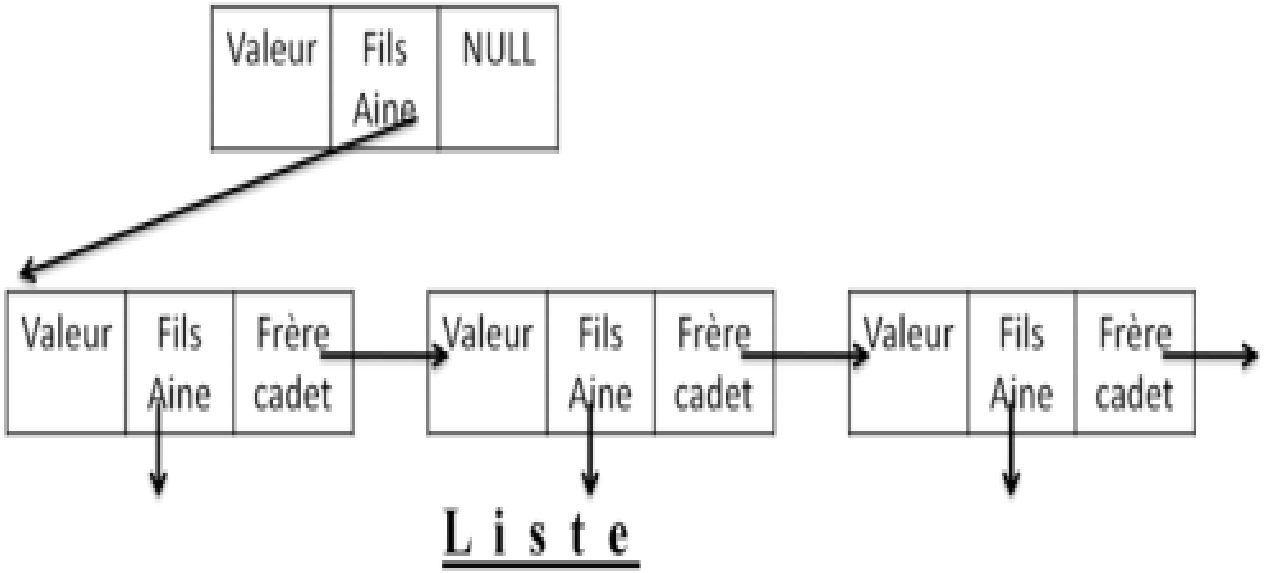
- **Listes parenthesées :**
  Prefixé: A (B(F) , C (G(L) , H) , D(I) , E(J, K(M, N)))
  Postfixé : ((F)B , ((L)G , H)C , (I)D , (J , (M , N)K)E)A

Arbre - **Implémentation**

- Les arbres peuvent être représentés par des tableaux, des listes non linéaires ou tous les deux
- Quand le nombre de fils de chaque élément est variable, on peut soit prévoir un tableau statique des adresses des fils, soit prévoir un tableau dynamique, ce qui optimise l'occupation mémoire mais complique l'ajout de fils supplémentaires.
- Pour avoir une gestion parfaitement dynamique, on peut créer une liste des adresses des fils

Arbre - **Implémentation**





Plutôt que de créer cette liste hors des nœuds, le plus simple (et qui utilise autant de mémoire) est d'associer à chaque nœud l'adresse de son fils aîné, et de son frère cadet. Accéder à tous les fils revient donc à accéder au fils aîné puis à tous ses frères

Arbre - Déclaration

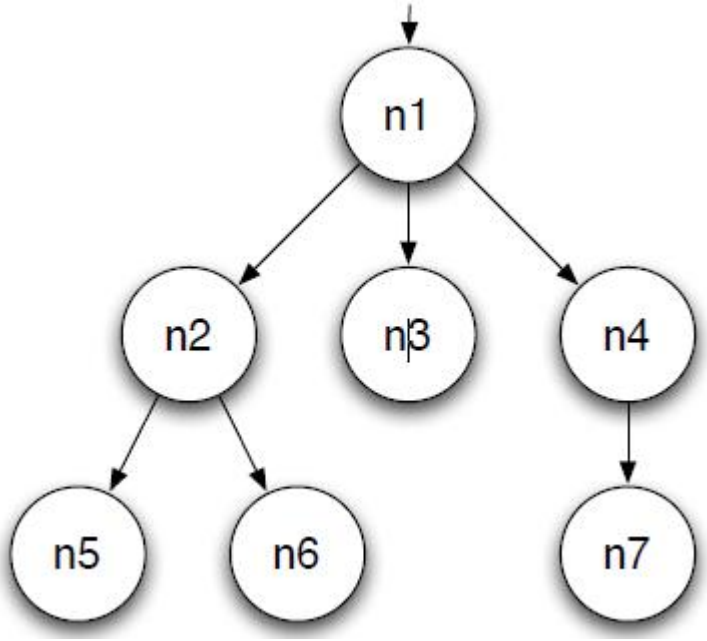
- Voici la déclaration d'un arbre général par chainage:

Type nœud = structure
Début
  Val : type_qc ;
  Fils_ainé : nœud;
  Frere_cadet : nœud ;
fin
arbre : pointeur :nœud ;

Code C:

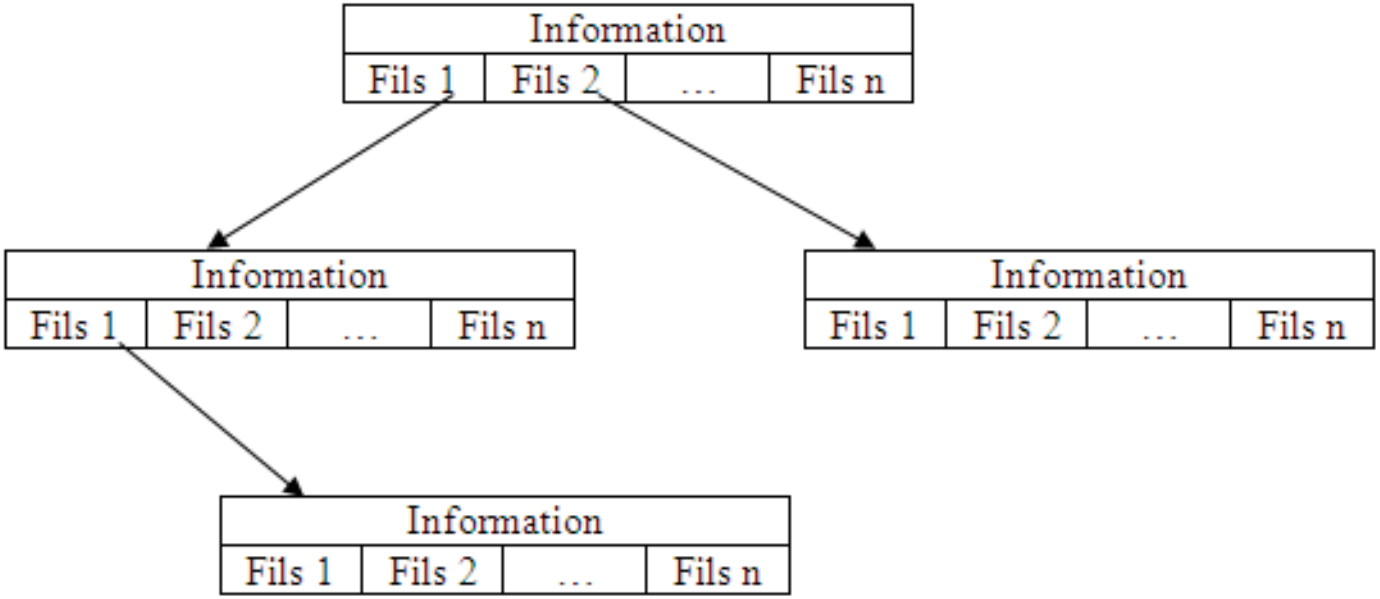
```
Typedef struct nœud
{
   Element   Val;
   nœud   *fils_aine;
   nœud   *frere_cadet;
}  *Arbre;
```

Exemple Implémentation statique



Num	Valeur	Fils 1	Fils 2	Fils 3
1	n1	2	3	4
2	n2	5	6	0
3	n3	0	0	0
4	n4	7	0	0
5	n5	0	0	0
6	n6	0	0	0
7	n7	0	0	0

Représentation dynamique




```
Type TNoeud = Structure
        Info : typeqq ;
        Fils : Tableau[1..NbFils] de Pointeur(TNoeud) ;
Fin ;
Var Racine : Pointeur(TNoeud) ;
```

Opérations sur l'arbre

Pour manipuler les structures de type arbre, on aura besoin des primitives suivantes :

- Allouer (N) : créer une structure de type TNoeud et rendre son adresse dans N.
- Liberer(N) : libérer la zone pointée par N.
- Aff_Val(N, Info) : Ranger la valeur de Info dans le champs info du nœud pointé par N.
- *Aff_Filsi(N1,N2) : Rendre N2 le fils numéro i de N1.*
- *Filsi(N) : donne le fils numéro i de N.*
- Valeur(N) : donne le contenu du champs info du nœud pointé par N.

Parcours des arbres

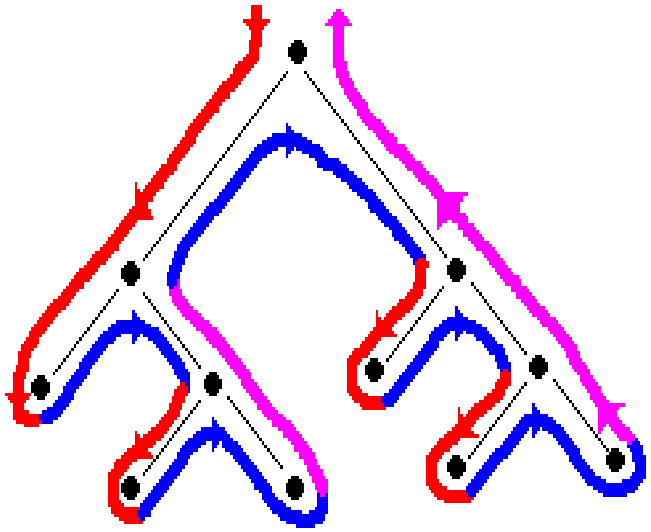
Le parcours d'un arbre consiste à passer par tous ses nœuds. Les parcours permettent d'effectuer tout un ensemble de traitement sur les arbres.

On distingue deux types de parcours :

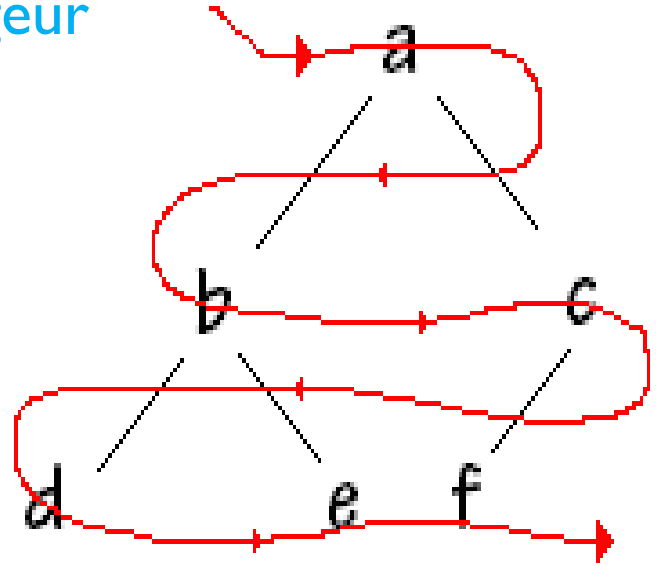
- Parcours en profondeur: on descend le plus profondément possible dans l'arbre puis, une fois qu'une feuille a été atteinte, on remonte pour explorer les autres branches
- Parcours en largeur **ou hiérarchique** : tous les nœuds à une profondeur i doivent avoir été visités avant que le premier nœud à la profondeur i + 1 ne soit visité. (de gauche à droite)

Schéma de parcours

- Parcours en profondeur



- Parcours en largeur



Parcours en profondeur

- on descend le plus profondément possible dans l'arbre puis, une fois qu'une feuille a été atteinte, on remonte pour explorer les autres branches en commençant par la branche "la plus basse" parmi celles non encore parcourues.
- L'algorithme récursif est le suivant :

```
Procédure PP( Noeud : Pointeur(TNoeud));
Début
    Si (Noeud ≠ Nil) Alors
        Pour i de 1 à NbFils faire
            PP(Filsi(Noeud))
        Fin Pour;
    Fin Si;
Fin;
```

Parcours en profondeur

- Le parcours en profondeur peut se faire en deux manière :
  - Parcours en profondeur Prefixe : où on affiche le père avant ses fils
  - Parcours en profondeur Postfixe : où on affiche les fils avant leur père.
- Les algorithmes récursifs correspondant sont les suivants :

Parcours en profondeur Prefixe

```
Procédure PPPrefixe( Noeud :
  Pointeur(TNoeud));
  Début
      Si (Noeud ≠ Nil) Alors
            Afficher(Valeur(Noeud)) ;
            Pour i de 1 à NbFils faire
                  PPPrefixe(Filsi(Noeud))
            Fin Pour;
      Fin Si;
  Fin;
```

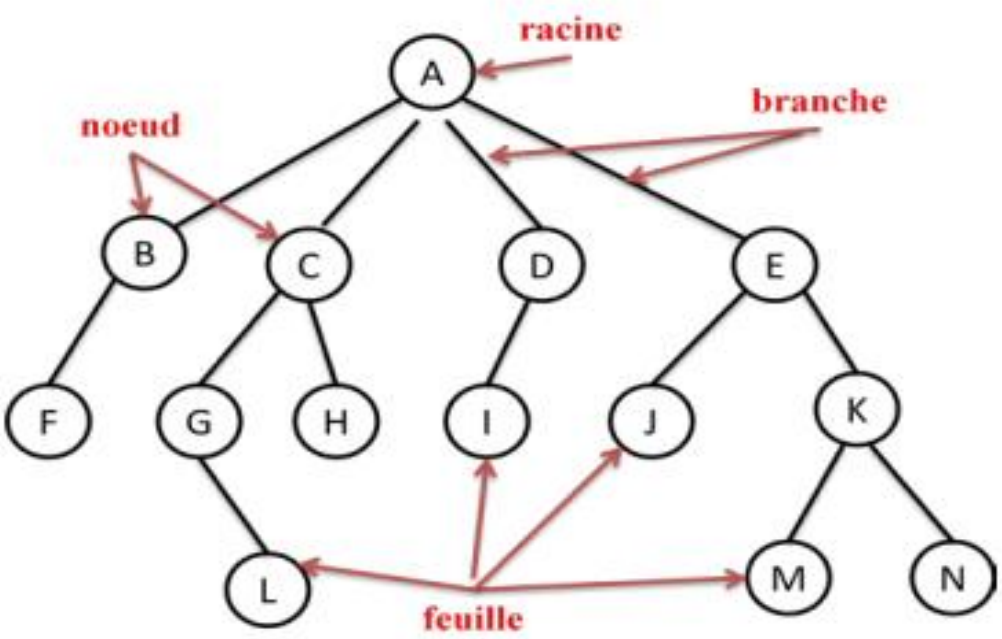
Parcours en profondeur Postfixe

```
Procédure PPPostfixe( Noeud : Pointeur(TNoeud));
  Début
      Si (Noeud ≠ Nil) Alors
            Pour i de 1 à NbFils faire
                  PPPostfixe(Filsi(Noeud))
            Fin Pour;
            Afficher(Valeur(Noeud)) ;
      Fin Si;
  Fin;
```

Exemple de parcours

Le parcours en profondeur préfixe de l'arbre suivant: ABF,CGL,H,DI,EJKMN

Le parcours en profondeur postfixe de l'arbre suivant: FBLGHCIDJMNKEA



Parcours en largeur

Dans un parcours en largeur, tous les nœuds à une profondeur i doivent avoir été visités avant que le premier nœud à la profondeur i + 1 ne soit visité. Un tel parcours nécessite que l'on se souvienne de l'ensemble des branches qu'il reste à visiter. Pour ce faire, on utilise une file d'attente.

```
Procédure PL( Noeud : Pointeur(TNoeud));
Var N : Pointeur(TNoeud) ;
Début
    Si (Noeud ≠Nil) Alors
            InitFile ; Enfiler(Neuud) ;
            Tant que (Non(FileVide)) faire
                    Défiler(N) ;
                    Afficher(Valeur(N)) ;
                    Pour i de 1 à NbFils faire
                            Si (Filsi(N) ≠ Nil) Alors
                                    Enfiler(Filsi(N)) ;
                            Fin Si;
                    Fin Pour;
            Fin TQ;
    Fin Si;
Fin;
```

Application sur l'arbre précédent: ABCDEFGHIJKLMN

Recherche d'un élément

```
Fonction Rechercher( Noeud : Pointeur(TNoeud) ;Val : Typeqq) : Booleen;
Var i : entier ;
Trouv : Booleen ;
Début
    Si (Noeud = Nil) Alors
            Rechercher ← Faux ;
    Sinon
            Si (Valeur(Noeud)=Val) Alors
                    Rechercher ← Vrai ;
            Sinon
                    i ← 1 ;
                    Trouv ← Faux ;
                    Tant que ((i ≤ NbFils) et non Trouv) faire
                            Trouv ← Rechercher(Filsi(Neoud),Val) ;
                            i ← i + 1;
                    Fin TQ;
                    Rechercher ← Trouv ;
            Fin Si;
    Fin Si;
Fin;
```

Calcul de la taille d'un arbre

```
Fonction Taille( Noeud : Pointeur(TNoeud)) : entier;
Var i,S : entier ;
Début
   Si (Noeud = Nil) Alors
         Taille ← 0 ;
   Sinon
         S ← 1 ;
         Pour i de 1 à NbFils faire
                  S ← S + Taille(Filsi(Noeud)) ;
         Fin Pour;
         Taille ← S ;
   Fin Si;
Fin;
```

Typologie des arbres (Cas particuliers)

En pratique, les arbres que l'on utilise, ou qui rendent efficaces les algorithmes proposés, sont des cas particuliers de la définition générale donnée précédemment.

- Arbre n-aire : un arbre n-aire d'ordre n est un arbre ou le degré maximum d'un nœud est égal à n.(en général pratique de limiter le nombre de fils qu'un nœud peut posséder)
- B-Arbre : Un arbre B d'ordre n est un arbre où :
  - la racine a au moins 2 fils
  - chaque nœud, autre que la racine, a entre n/2 et n fils
  - tous les nœuds feuilles sont au même niveau
- Arbre binaire : c'est un arbre ou le degré maximum d'un nœud est égal à 2.
- Arbre binaire de recherche (ABR) : c'est un arbre binaire où la clé de chaque nœud est supérieure à celles de ses descendants gauche, et inférieure à celles de ses descendants droits.